

Trabalho Intermediário

Victor Marcius Magalhães Pinto

1 Descrição da Tarefa

teste

2 Implementação de funções intermediárias

Para a execução da tarefa, algumas funções auxiliares foram implementadas. A implementação das mesmas pode ser visto a seguir.

```
>  
> #test  
>
```

3 Execução do código

Primeiramente, foram carregadas as imagens contidas no pacote de faces. O pacote contém 40 classes de imagens, com 10 amostras cada.

3.1 Carregando os dados

```
> data( faces )  
> faces <- t( faces )  
> rotate <- function(x) t( apply(x, 2, rev) )  
> y <- NULL  
> for(i in 1:nrow(faces) )  
+ {  
+   y <- c( y, ((i-1) %/% 10) + 1 )  
+ }  
> # Nomeando os atributos  
> nomeColunas <- NULL  
> for(i in 1:ncol(faces) )  
+ {  
+   nomeColunas <- c(nomeColunas, paste("a", as.character(i), sep=".") )  
+ }  
> nomeLinhas <- NULL  
> for(i in 1:nrow(faces)) {
```

```

+     nomeLinhas <- c(nomeLinhas, paste("face", as.character(y[i]), as.character(i)))
+ }
> colnames(faces) <- nomeColunas
> faces <- as.data.frame(faces, row.names = nomeLinhas)
> rm(nomeColunas)
> rm(nomeLinhas)

```

Cada imagem foi transformada, de uma matrix de 64x64 pixels, em um vetor de 4096 features, e inserida em um dataframe de 400 linhas.

3.2 Diminuindo o número de atributos com PCA

Em seguida, foi realizada a diminuição do número de atributos da base de dados utilizando o algoritmo do PCA. Em implementações anteriores, foi utilizada a função *preProcess* juntamente com a *predict* do pacote *caret*, o que nos informava o número mínimo de atributos para uma exatidão de 95% de 123 features. Porém, o pacote apresentou problemas de funcionamento em dias posteriores, o que nos exigiu a utilização da função *prcomp*, que pode ser vista no trecho de código a seguir.

```

> facesPCAaux <- prcomp(faces, center=TRUE, retx=TRUE, scale=TRUE)
> facesPCA <- facesPCAaux$x[,1:5]
>

```

Observando a implementação da função do classificador de Bayes, e da geração de pdfs para amostras multivariáveis, uma matriz de correlação invertível é gerada apenas se o número de linhas for menor ou igual ao número de colunas, única possibilidade que não gera uma matriz singular quando da execução da função *solve*. Como são 10 amostras de cada classe, e para o trabalho será utilizada 5 amostras para treino e 5 para testes, o número de features máximo que pode ser utilizado, para o correto funcionamento do algoritmo do classificador de Bayes, é de 5 features.

3.3 Diminuindo o número de atributos com MDS

O mesmo procedimento, e a mesma limitação do número de features foi realizado para o método MDS. O algoritmo pode ser visto a seguir.

```

> kMDS <- 5
> facesMDS <- cmdscale(dist(faces), k=kMDS)
>

```

3.4 Gerando sets de treino e teste

Como dito anteriormente, para o treinamento e classificação pelos algoritmos, foram utilizadas 5 amostras de cada face. A geração das amostras pode ser vista no trecho de código a seguir. A mesma possui uma alta complexidade de implementação, visto da utilização de loops internos, e de toda o algoritmo, é o trecho que consome mais tempo de execução

```
> dim_classe <- 10
> numClasses <- 400
> numAmostras <- 10
> seqN <- sample(numAmostras)
> porcAmostrTrain <- 0.5
> N <- seqN[1:(porcAmostrTrain*numAmostras)]
> nSamplesTrain <- length(N)
> n <- seqN[(porcAmostrTrain*numAmostras+1):numAmostras]
> nSamplesTest <- length(n)
> xtreino <- c()
> xtreinoPCA <- c()
> xtreinoMDS <- c()
> ytreino <- c()
> xteste <- c()
> xtestePCA <- c()
> xtesteMDS <- c()
> yteste <- c()
> for(r in seq(1,numClasses,numAmostras)) {
+   for(i in N) {
+     xtreino <- rbind(xtreino, (faces[r+i-1,]))
+     xtreinoPCA <- rbind(xtreinoPCA, (facesPCA[r+i-1,]))
+     xtreinoMDS <- rbind(xtreinoMDS, (facesMDS[r+i-1,]))
+     ytreino <- c(ytreino, (y[r+i-1]))
+   }
+
+   for(i in n) {
+     xteste <- rbind(xteste, (faces[r+i-1,]))
+     xtestePCA <- rbind(xtestePCA, (facesPCA[r+i-1,]))
+     xtesteMDS <- rbind(xtesteMDS, (facesMDS[r+i-1,]))
+     yteste <- c(yteste, (y[r+i-1]))
+   }
+ }
>
```